Solving the maximum clique problem with a hybrid algorithm

Derek H. Smith* and Stephanie Perkins

Division of Mathematics and Statistics, University of South Wales ,
Pontypridd, CF37 1DL, Wales, UK
E-mail: derek.smith@southwales.ac.uk
E-mail: stephanie.perkins@southwales.ac.uk
*Corresponding author

Roberto Montemanni

Dalle Molle Institute for Artificial Intelligence (IDSIA - USI/SUPSI)
Galleria 2, 6928 Manno, Switzerland
E-mail: roberto.montemanni@supsi.ch

Abstract: A hybrid algorithm for the maximum clique problem is presented. A heuristic is used to generate cliques and these are improved by some simple optimizations and tabu search. All components of the algorithm make use of a pseudoexact algorithm, which is an exact algorithm with some specialized pruning. Preprocessing is useful for some instances. The algorithm is shown to be successful using standard and new benchmarks.

Keywords: combinatorial optimization; maximum clique; hybrid algorithm; tabu search; pseudoexact algorithm; preprocessing; benchmarks.

Reference to this paper should be made as follows: Smith, D.H., Perkins, S. and Montemanni, R. (20xx) 'Solving the maximum clique problem with a hybrid algorithm', *Int. J. of Metaheuristics*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Derek H. Smith is Emeritus Professor of Mathematics at the University of South Wales (formerly the University of Glamorgan), where he worked from 1971 to 2014. He has B.Sc. and Ph.D. degrees from the University of Southampton, U.K. and a D.Sc. degree from the University of Glamorgan. His research interests include metaheuristic algorithms, radio frequency assignment, graph theory, coding theory and data compression.

Stephanie Perkins received her B.Sc. degree from the University of North Wales, Bangor, U.K., and her Ph.D. degree from Royal Holloway, University of London, London, U.K. She is Head of Mathematics at the University of South Wales (formerly the University of Glamorgan), which she joined in 1994. Her research interests include coding theory, data compression, and combinatorics.

Roberto Montemanni is Professor of Advanced Algorithms at the University of Applied Sciences of Southern Switzerland. He has a PhD from the University of Glamorgan, U.K. and a Laurea degree in computer science from the University of Bologna, Italy. His main research interests include the development of mathematical programming models and algorithms for combinatorial optimization problems arising in transportation and telecommunication.

1 Introduction

In this paper an algorithm for the maximum clique problem is presented. It is shown that a combination of some straightforward ideas leads to an effective method, able to match state-of-the-art results. For most instances it is slower than the best current algorithms, although for a few instances it is faster. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. A clique C of G is a subset of the vertices of V with every pair of vertices of C adjacent. A maximum clique is a clique with the maximum number of vertices. Problems involving cliques arise in many applications including bioinformatics, examination planning, location problems, signal transmission analysis, and social network analysis (see the survey (Wu and Hao, 2015) for references). Other applications of the maximum clique problem arise in radio frequency assignment (see, for example (Leese and Hurley, 2002)) and in the construction of various types of error-correcting code with the maximum number of codewords (Sloane, 2015; Smith and Montemanni, 2012).

As the maximum clique problem is NP-hard, heuristic methods are necessary for larger problems. The recent paper (Wu and Hao, 2015) surveys exact and heuristic algorithms for the maximum clique problem. Several heuristic algorithms use tabu search (Gendreau, 1993; Wu et al., 2012), including most notably the recent algorithm of Jin and Hao (2015) which matches the best known results for all instances attempted. The reader is referred to (Wu and Hao, 2015) for extensive further references. The interest of the current authors in the maximum clique problem arose from the construction of codes with the maximum number of codewords. Thus the emphasis here is on finding the largest possible clique in a practical run time rather than in achieving the fastest possible algorithm. In this respect the algorithm is successful, finding the best known result for almost all instances attempted. The benchmark instances include 15 arising from code construction problems as well as 9 hard DIMACS instances (1993a;1993b) and 15 BHOSLIB instances (2015) to allow comparison with other algorithms.

The algorithm HTS (Hybrid Tabu Search) described here is a hybrid made up of a main heuristic that generates good starting solutions, a number of simple optimizations and tabu search. The tabu search differs from many other tabu search algorithms in that (i) it makes use of a pseudoexact algorithm, and (ii) vertices are only added to the current clique after a defined number of vertices have been removed, unless a new best clique can be obtained.

Section 2.1 describes the general structure of the algorithm. Section 2.2 describes the exact algorithm and the modification making it pseudoexact. Section 2.3 describes the main heuristic and Section 2.4 describes the simple i_optimizations that can be applied. Section **??** describes the tabu search. Section **??** describes the instances used to test the algorithm and Section **??** describes some preprocessing that has been found useful for some of the benchmarks. Parameter tuning and results are described in Section **??** and the performance of the algorithm is summarized in Section **??**.

2 Algorithm

2.1 The general structure of the algorithm HTS

The structure of the algorithm for finding a clique $BestC$ is outlined in Algorithm 1. Within an outer while loop the main heuristic generates a current clique and four thresholds θ_1, θ_2,

θ_3, θ_{ts} are used to ensure that various optimizations are only applied to promising current solutions.

Algorithm 1: Outline of the overall structure of the algorithm :
$BestC := \emptyset$;
$k := 0$
while $runtime \leq max\_runtime$ **do**
  $k := k + 1$
  $C := Main\_heuristic(k)$
  if $|C| \geq \theta_1$ **then** $C := 1\_optimize(C)$; **end if**
  if $|C| \geq \theta_{\mathrm{ts}}$ **then** $C := ts\_optimize(C)$; **end if**
  if $|C| \geq \theta_2$ **then** $C := 2\_optimize(C)$; **end if**
  if $|C| \geq \theta_3$ **then** $C := 3\_optimize(C)$; **end if**
  if $|C| > |BestC|$ **then** $BestC := C$; **end if**
end while

2.2 Pseudoexact search

Pseudoexact search is used in the main heuristic, all i_optimizations and in tabu search. The pseudoexact search is a version of the exact algorithm of Carraghan and Pardalos (1990), with an additional pruning condition. This condition makes it unnecessary to implement the various enhancements to the algorithm that have been proposed.

Algorithm 2: An exact algorithm applied to a set of vertices V:
Require *A set of selected vertices $F \subseteq V$ and a set of potential expansion vertices $S \subseteq V$. The best clique retrieved so far is contained in the external set $Best$ and $External\_lower\_bound$ is a supplied external value. The recursive algorithm is invoked with $F := \emptyset$, $S := V$ and $Best := \emptyset$.*
Exact(**F**,** S**)
while $S \neq \emptyset$ **do**
  select $s \in S$;
  $S := S \backslash \{s\}$;
  $S' := S$;
  for $z \in S'$ **do**
    if$(z, s) \notin E$ **then**
      $S' := S' \backslash \{z\}$;
    end if
  end for
  $F' := F \cup \{s\}$;
  if $S' = \emptyset$ *and* $|F'| > |Best|$ **then**
    $Best := F'$;
  else
    if $|F'| + |S'| > \max\{|Best|, External\_lower\_bound - 1\}$ **then**
      Exact(F'*,* S'*);*
    end if
  end if
end while

The usual condition is that pruning takes place unless the number of vertices in the current clique plus the number of vertices under consideration adjacent to all these vertices be greater than the size of the best clique found so far. Here an extra condition is used involving an external lower bound. Essentially pruning takes place if an improvement to the overall solution of the current optimization cannot be found. The underlying exact algorithm presented here as Algorithm 2 is a recursive algorithm, although the actual implementation was non-recursive for efficiency reasons. Although Algorithm 2 is an exponential time algorithm, it often works well in practice. An enhancement to the pruning described can be determined by a short initial run, recording the size of all subproblems solved by exact search and the size of the clique found. An Excel chart is constructed for the instance. For small subproblems the original pruning is used, but for larger subproblems it may be observed that the size of the clique found is always significantly less than the size of the subproblem. A linear relationship (chosen conservatively) can then be used for pruning. Thus the test $|F'| + |S'| > \max\{|Best|, External\_lower\_bound - 1\}$ might typically be replaced by $|F'| + (|S'|/2) + 5 > \max\{|Best|, External\_lower\_bound - 1\}$. This does speed the algorithm significantly without affecting the quality of solutions generated, although the guarantee that an exact solution is found is lost for larger subproblems. The precise linear function used for each instance will be described in Section **??**. The pseudoexact algorithm is then denoted Pseudoexact(F, S).

2.3 The main heuristic

The main heuristic at iteration k is presented as Algorithm 3. For any current clique C, $N(C)$ denotes the set of vertices of V adjacent to all vertices in C. A sequence of values $S = [s_i]$ is defined, typically $S = [1, 50, 80, 100, 120, 160, 200, 300, 600, 800]$. The variable $adjchoices$ in the main heuristic cycles through the values in S and controls the choice of the next vertex in $N(C)$ to add to the clique C. The exact choice of S is not critical as long as a spread of values up to the maximum value of $|N(C)|$ is included. A set U consists of all vertices of $N(C)$ if $|N(C)| \leq adjchoices$, or $adjchoices$ randomly selected vertices of $N(C)$ otherwise. A vertex u of U is chosen to add to C which gives the largest value of $N(C \cup \{u\})$. Alternatively, if $|N(C)|$ is less than a threshold λ_2 the pseudoexact algorithm is applied to $N(C)$ and the union of C and the best clique found is taken. An enhancement to the algorithm is to store both C and $N(C)$ the first time that $|N(C)|$ is less than a larger threshold λ_1. The pseudoexact algorithm is applied to the stored $N(C)$ at the end of the algorithm (and used in the same way) only if the number of vertices in current clique C is at least θ_1. In this way a larger clique is sometimes obtained. The rationale for this enhancement is that the pseudoexact algorithm inevitably takes longer in this circumstance, so should only be applied in promising situations. The run time for a single iteration of Algorithm 3 is approximately linear in s, but the inclusion of larger values of s in S improves the performance of the algorithm significantly.

2.4 Various i_optimizations

i_optimizations were implemented for $i = 1, 2, 3$. Different thresholds were used for the three optimizations because of their different speeds. Essentially i_optimization considers all possible subsets of the current clique C with i vertices. Each subset S is removed in turn and the pseudoexact algorithm is applied to $N(C \setminus S)$ to find a best clique $Best$. If an improved clique $(C \setminus S) \cup Best$ is found then i_optimization is applied to the new clique.

Algorithm 3: Main heuristic at iteration k:
Main_heuristic(k)
C :=*{random vertex in V} (or C := starting clique as in Section* **??***)*
$Pseudoexactstored := False$; $adjchoices := s_{1+k \bmod |S|}$
while $|N(C)| > 0$ **and** $|N(C)| \geq \theta_1 - |C|$ **do**
 if $|N(C)| < \lambda_2$ **then**
  $Best := \emptyset$; $External\_lower\_bound := \theta_1 - |C|$
  Pseudoexact($\emptyset$, $N(C)$)
  $C := C \cup Best$
 else
  if $|N(C)| < \lambda_1$ **and** $Pseudoexactstored = False$ **then**
   $C_{\text{stored}} := C$; $N(C)_{\text{stored}} := N(C)$
   $Pseudoexactstored := True$
  end if
  if $|N(C)| \leq adjchoices$ **then**
   $U := N(C)$
  else
   $U := \{v_{j_1}, v_{j_2}, \ldots, v_{j_{adjchoices}} | v_{j_i} \in N(C)$ selected randomly$\}$
  end if
  Choose $u \in U$ such that $|N(C \cup \{u\})|$ is maximal
  $C := C \cup \{u\}$
 end if*;*
end while
if $Pseudoexactstored = True$ **and** $|C| \geq \theta_1$ **then**
 $Best := \emptyset$; $External\_lower\_bound := \theta_1 - |C_{\text{stored}}|$
 Pseudoexact($\emptyset$, $N(C)_{\text{stored}}$)
 $C_{\text{temp}} := C_{\text{stored}} \cup Best$
 if $|C_{\text{temp}}| > |C|$ **then** $C := C_{\text{temp}}$ **end if**
end if*;*

Algorithm 4: i_Optimize :
The best clique retrieved so far is contained in the external set $Current\_best\_opt$. The recursive algorithm is invoked with $Current\_best\_opt := C$. Following the call to the procedure, $C := Current\_best\_opt$.
i_Optimize(C)
for *all $S \subset C$ with $|S| = i$* **do**
 if $|N(C \setminus S)| > i$ **then**
  $Best := \emptyset$; $External\_lower\_bound := i + 1$
  Pseudoexact($\emptyset$, $N(C \setminus S)$)
  if $|Best| > i$ **then**
   $Current\_opt := Best \cup (C \setminus S)$
   if $Current\_opt > Current\_best\_opt$ **then**
    $Current\_best\_opt := Current\_opt$
    i_Optimize($Current\_opt$)
   end if
  end if
 end if
end for

1_optimization is fast and particularly useful to find good starting solutions for tabu search. 2_optimization and 3_optimizations were useful much less frequently, and were not used for any of the best results presented here.

2.5 Tabu search

The tabu search algorithm ts_Optimize is presented as Algorithm **??**. The parameters of the algorithm are $tstime$, $tstenure$ and $tssetmin$, specifying the maximum run time in seconds with no improvement, the length of the tabu list and the minimum size for the current clique $tscurrent$. At each iteration, each vertex v in $tscurrent$ is considered for removal. A maximum clique in the neighbour set $N(tscurrent \setminus \{v\})$ is found by the pseudoexact algorithm. A list L_1 is drawn up of the vertices for which the union of $tscurrent \setminus \{v\}$ and the best clique found in $N(tscurrent \setminus \{v\})$ is maximal, with $|N(tscurrent \setminus \{v\})|$ also required to be maximal as a secondary condition to break some of the ties and guide the search towards further improvement. This list contains vertices which might meet the aspiration criterion (give the largest clique so far found by ts_Optimize). A similar list L_2 is drawn up where only nontabu vertices of $N(tscurrent \setminus \{v\})$ are considered. If the aspiration criterion is met, a vertex of L_1 is selected at random and the new best clique becomes the current clique. Otherwise a vertex of L_2 is selected at random and removed from the current clique. It is also added to the tabu list, with the oldest vertex in the tabu list removed if the list length would exceed $tabutenure$. If after removing a vertex of L_2 the size of the current clique $tscurrent$ is at most $tssetmin + 1$ and there are less than 30 nontabu vertices in the neighbour set $N(tscurrent)$ then $tscurrent$ is augmented by the relevant maximum clique of nontabu vertices in the neighbour set $N(tscurrent)$. If there are at least 30 nontabu vertices in the neighbour set $N(tscurrent)$ then a vertex is randomly selected to augment $tscurrent$. This last step is simply a protection mechanism to avoid the pseudoexact algorithm attempting to solve problems that would take too long. Normally the values of $tstenure$ and $tssetmin$ are chosen so this step never occurs, or only occurs very rarely.

3 Benchmark Instances

As the interest of the authors in maximum clique problems arose from problems in coding theory, many of the instances are taken from permutation code problems and other coding theory problems. The instances (11,5), (12,4), (12,5), (10,4) arose in (Smith and Montemanni, 2012) and were the harder instances that arose in that work. The instance (7,5) is an attempt to find a maximal (7,5) permutation code directly without using a group. The vertices correspond to the permutations on 7 symbols, excluding those permutations at Hamming distance 1, 2, 3 or 4 from the identity permutation. Two vertices are adjacent if they are at Hamming distance ≥ 5. Also selected were maximum clique versions of Sloane's Coding theory challenge problems from (Sloane, 2015). Specifically, the problems with 1024 and 2048 vertices were selected: 1dc.1024, 1dc.2048, 1et.1024, 1et.2048, 1tc.1024, 1tc.2048, 1zc.1024, 1zc.2048, 2dc.1024, 2dc.2048. The 9 DIMACS instances (1993a; 1993b) referred to in (Wu and Hao, 2015) as "the hard DIMACS instances" were selected to allow comparison with other work: brock400_2, brock400_4, brock800_2, brock800_4, C2000_9, C4000_5, keller6, MANN_a45, MANN_a81. Finally, 15 BHOSLIB instances (BHOSLIB, 2015) with 450, 595 and 1150 vertices were used to show that much of the parameter tuning is not particularly critical.

Algorithm 5: Tabu search :
ts_Optimize(C***)***
$Ts\_Starttime := Timenow$; $Ts\_Endtime := Ts\_Starttime$
$Tabulist := \emptyset$; $Tsimprove := True$; $Tscurrent := C$; $Tsbest := C$
while ***(***$(Ts\_Endtime - Ts\_Starttime < Tstime$ *or* $Tsimprove = True)$***)*** **do**
  $Tsimprove := False$; $Aspiration := False$
  for *all* $v \in Tscurrent$ **do**
    $Best := \emptyset$; $External\_lower\_bound := |Tsbest| - |Tscurrent| + 1$
    Pseudoexact($\emptyset$, $N(Tscurrent \setminus \{v\})$*)*
    $Tstemp1 := Best \cup (Tscurrent \setminus \{v\})$
    Update a list L_1 *of all* $v \in Tscurrent$ *with* $|Tstemp1|$ *maximal (and for this value of* $|Tstemp1|$ *the value* $|N(Tscurrent \setminus \{v\})|$ *is maximal)*
    if $|Tstemp1| > |Tsbest|$ **then** $Aspiration := True$ **else**
      $Nontabu := \{w \in N(Tscurrent \setminus \{v\}) | w \notin Tabulist\}$
      $Best := \emptyset$;$External\_lower\_bound := 0$
      Pseudoexact($\emptyset$, $Nontabu$*)*
      $Tstemp2 := Best \cup (Tscurrent \setminus \{v\})$
      Update a list L_2 *of all* $v \in Tscurrent$ *with* $|Tstemp2|$ *maximal (and for this value of* $|Tstemp2|$ *the value* $|Nontabu|$ *is maximal)*
    end if
  end for
  if *Aspiration = true* **then** *select* $v' \in L_1$ *randomly;* $Ts\_Starttime := Timenow$
    $Best := \emptyset$; $External\_lower\_bound := |Tsbest| - |Tscurrent| + 1$
    Pseudoexact($\emptyset$, $N(Tscurrent \setminus \{v'\})$*)*
    $Tscurrent := Best \cup (Tscurrent \setminus \{v'\})$; $Tsbest := Tscurrent$
  else
    Select $v'' \in L_2$ *randomly*
    $Tscurrent := Tscurrent \setminus \{v''\}$
    Add v'' *to* $Tabulist$ *(removing oldest entry if list length exceeds* $Tstenure$*)*
    if $|Tscurrent| \leq Tssetmin + 1$ **then**
      $Nontabu := \{w \in N(Tscurrent) | w \notin Tabulist\}$
      if $|Nontabu| < 30$ **then**
        $Best := \emptyset$; $External\_lower\_bound := 0$
        Pseudoexact($\emptyset$, $Nontabu$*)*
        $Tscurrent := Best \cup Tscurrent$
      else
        Select $v''' \in Nontabu$ *randomly*
        $Tscurrent := \{v'''\} \cup Tscurrent$
      end if
  end if
  $Ts\_Endtime := Timenow$
end while
if $|Tsbest| > |C|$ **then** $C := Tsbest$ **end if**

4 Starting Vertices and Other Problem Simplifications

For some graphs it is possible to derive starting vertices that are automatically in some maximum clique. A vertex adjacent to all other vertices can always be chosen as a starting vertex. Similarly, for a vertex transitive graph any vertex may be chosen as a starting vertex, and for a distance transitive graph it may be possible to derive two starting vertices. Such starting vertices are not always helpful, but are helpful for the MANN problems studied here. Jin and Hao (2015) state that typical maximum clique algorithms have particular difficulties in solving these two instances.

Consider the graph G of the instance MANN_a45. A Magma graph command "*DegreeSequence*" (Magma Handbook, 2017) indicates that there is a set S of 45 vertices (numbered from 1 to 45) of degree 1012 and the remaining 990 vertices form a set T of vertices of degree 1031. The algorithm presented here will easily find a clique of size 343 containing 13 vertices of S and a clique of size 344 containing 14 vertices of S. This suggests that a clique of size 345 will contain 15 vertices of S.

Magma graph commands indicate that the graph induced by S is complete. For vertex v_i in S define the set N_i to contain the vertices of T not adjacent to v_i. Then Magma again indicates that the sets N_i are disjoint, $|N_i| = 22$ for all $v_i \in S$, the graph induced by N_i is complete for each i, and for each pair of sets N_i, N_j there is precisely one pair of non adjacent vertices $v' \in N_i$, $v'' \in N_j$. These 990 edges of the complement $\overline{G}_T$ of the graph G_T induced by T fall into 330 disjoint triangles.

For any 15 vertices $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{15}}$ of S consider the distribution of these 330 triangles in $\overline{G}_T$. Let $U = \cup_{j=1}^{15} N_{i_j}$ and G_U be the subgraph of G induced by U. Denote by a the number of triangles with all 3 vertices in U, by b the number of triangles with precisely 1 vertex in U, by c the number of triangles with precisely 2 vertices in U and by d the number of triangles with all three vertices in $T \setminus U$. Counting (i) the 330 vertices in a maximum clique in the graph $G_{T\setminus U}$, (ii) the number of edges in $\overline{G}_U$, (iii) the number of pairs of vertices $u \in U$, $v \in T \setminus U$ that are not adjacent, (iv) the number of edges in $\overline{G}_{T\setminus U}$, (v) the total number of triangles, gives equations:

$$\begin{aligned} b + c + d &= 330 \\ 3a + c &= 105 \\ 2b + 2c &= 450 \\ b + 3d &= 435 \\ a + b + c + d &= 330 \end{aligned}$$

with solution $a = 0$, $b = 120$, $c = 105$ and $d = 105$.

This suggests that $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{15}}$ should be chosen so that G_U contains no triangles. For each triangle with vertices in the sets $N_{i_k}, N_{i_l}, N_{i_m}$ the triangle will be labelled by (and said to contain) the vertices $v_{i_k}, v_{i_l}, v_{i_m}$. Finding 30 vertices of S such that each of the 330 triangles contains at least one of these vertices is not quite trivial (greedy selection failed), but can be done as follows. Partition the 45 vertices of S into 9 classes $C_1 = \{v_1, v_2, \ldots, v_5\}$, $C_2 = \{v_6, v_7, \ldots, v_{10}\}$, $C_3 = \{v_{11}, v_{12}, \ldots, v_{15}\}$, $C_4 = \{v_{16}, v_{17}, \ldots, v_{20}\}$, $C_5 = \{v_{21}, v_{22}, \ldots, v_{25}\}$, $C_6 = \{v_{26}, v_{27}, \ldots, v_{30}\}$, $C_7 = \{v_{31}, v_{32}, \ldots, v_{35}\}$, $C_8 = \{v_{36}, v_{37}, \ldots, v_{40}\}$, $C_9 = \{v_{41}, v_{42}, \ldots, v_{45}\}$. Considering all choices of 6 classes and checking that every triangle contains at least one vertex of the 6 classes gives 9 solutions of this type. Choosing the solution with classes

C_2, C_3, C_4, C_5, C_8, C_9 for the non-selected vertices of S gives selected vertices of S: v_1,v_2, v_3, v_4,v_5, v_{26}, v_{27},v_{28}, v_{29}, v_{30}, v_{31}, v_{32}, v_{33},v_{34}, v_{35}. Using these 15 vertices of S as starting vertices, the main heuristic gives a clique of MANN_a45 with 345 vertices immediately. It is known that 345 is an optimal solution.

Now consider the graph G of the instance MANN_a81. A Magma graph command indicates that there is a set S of 81 vertices (numbered from 1 to 81) of degree 3280 and the remaining 3240 vertices form a set T of vertices of degree 3317. Our heuristic will easily find a clique of size 1098 containing 18 vertices of S. This suggests that a clique of size 1100 will contain 20 vertices of S.

Magma graph commands again indicate that the graph induced by S is complete, the sets N_i are disjoint and that $|N_i| = 40$ for all $v_i \in S$. The graph induced by N_i is complete for each i, and for each pair of sets N_i, N_j there is precisely one pair of non adjacent vertices $v' \in N_i$, $v'' \in N_j$. These 3240 edges of the complement $\overline{G}_T$ of the graph G_T induced by T fall into 1080 disjoint triangles.

For any 20 vertices $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{20}}$ of S consider the distribution of these 1080 triangles in $\overline{G}_T$. Let $U = \cup_{j=1}^{20} N_{i_j}$ and G_U be the subgraph of G induced by U. A similar set of equations to those for MANN_a45

$$\begin{aligned} b + c + d &= 1080 \\ 3a + c &= 190 \\ 2b + 2c &= 1220 \\ b + 3d &= 1830 \\ a + b + c + d &= 1080 \end{aligned}$$

shows that a set of vertices $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{20}}$ should be found so that G_U contains no triangles. An integer linear program finds an optimal hitting set of 61 vertices of S such that each of the triangles contains at least one of these 61 vertices. For each vertex v_i in S there is a corresponding 0-1 variable c_i. For each triangle there is an inequality of the form $c_{i_k} + c_{i_l} + c_{i_m} \geq 1$ and the objective function to be minimized is $\sum_{i=1}^{81} c_i$. The associated integer linear program is solved to optimality by the 6.0 solver (Gurobi, 2017). The computation time is below 3 seconds on a 2.0GHz Intel Core i7 computer. Excluding the 61 vertices of the hitting set, the remaining 20 vertices of S form a set of starting vertices for our heuristic: v_3, v_6, v_{10}, v_{11}, v_{13}, v_{17}, v_{21}, v_{27}, v_{30}, v_{36}, v_{38}, v_{40}, v_{50}, v_{52}, v_{59}, v_{61}, v_{64}, v_{71}, v_{75}, v_{78}. Using these 20 vertices of S as starting vertices, the main heuristic gives a clique of MANN_a81 with 1100 vertices immediately.

The above argument can be extended to show that 1100 is an upper bound for MANN_a81, a result that was first found computationally in (McCreesh and Prosser, 2013), taking 31 days using 24 threads on a 12-core hyper-threaded dual 2.4GHz Intel Xeon E5645 computer. As $\overline{G}_T$ consists of 1080 disjoint triangles, a clique in G_T cannot contain more than 1080 vertices. Thus for a clique in G to contain more than 1100 vertices it must contain more than 20 vertices of S. Specifically, suppose that a clique of size $1100 + e$, ($e \geq 1$) contains $20 + j$ vertices of S, ($j \geq e$). Denote these $20 + j$ vertices by $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{20+j}}$. Let $U = \cup_{l=1}^{20+j} N_{i_l}$ and G_U be the subgraph of G induced by U. Denote by a the number of triangles with all 3 vertices in U and by b the number of remaining triangles. The first and last equations above become:

$$\begin{aligned} b + c + d &= 1080 + e - j \\ a + b + c + d &= 1080 \end{aligned}$$

so $a = j - e$. Only $1080 - j + e$ triangles contain at least one of the set $V = \{S \setminus \{v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{20+j}}\}\}$ of the remaining $61 - j$ vertices of S. For each of the $j - e$ triangles containing no vertex of V, choose one vertex label and let W be the set of $j - e$ vertices chosen. Then $V \cup W$ would be a hitting set for the 1080 triangles with $61 - j + j - e = 61 - e$ vertices ($e \geq 1$). This is impossible as the integer linear program shows that 61 is the minimum size of a hitting set for the 1080 triangles.

A different problem simplification arises for the instances 1et.1024, 1et.2048, 1tc.1024, 1tc.2048. Two Magma commands show that the complement graphs have 11, 12, 11 and 12 connected components respectively. Construct the subgraphs of the original graphs induced by the vertices of these components. The union of the vertices of cliques in these subgraphs then induce a clique in the original graph. This simplification was not used for 1et.1024 and 1tc.1024, but was used for 1et.2048 and 1tc.2048 to reduce run times.

5 Results

Table 1 Tuning parameters for instances that do not require tabu search to find the best solution

Instance	θ_1	θ_2	θ_3	λ_1	λ_2	C&P	$adjchoices$
(11,5) (Smith & Montemanni 2012)	26	27	27	105	85	A	S
(12,4) (Smith & Montemanni 2012)	212	221	221	60	45	B	S
(12,5) (Smith & Montemanni 2012)	25	26	26	105	85	A	S
brock400_2 (DIMACS 1993b)	26	26	26	105	85	A	ONE
brock400_4 (DIMACS 1993b)	25	31	31	105	45	A	ONE
brock800_2 (DIMACS 1993b)	21	21	21	180	85	A	ONE
brock800_4 (DIMACS 1993b)	21	21	21	105	85	A	ONE
MANN_a45 (DIMACS 1993b)	344	344	346	50	50	A	S
MANN_a81 (DIMACS 1993b)	1100	1101	1101	45	45	A	S
2dc.1024 (Sloane 2015)	14	14	14	90	80	A	S

All instances were solved on Intel Pentium 4 3.40 GHz or Celeron (R) 2.79 GHz processors with 4 GB of RAM. The tuning parameters for instances that do not require tabu search to find the best solution are shown in Table **??** and for instances that do require tabu search they are shown in Table **??**. Instances 1et.2048 and 1tc.2048 are split into subproblems as described in Section **??** and so are excluded from Table **??**.

The parameters θ_1, θ_2, θ_3 and θ_{ts} are the thresholds associated with the application of 1_optimization, 2_optimization, 3_optimization and tabu search respectively, as in Algorithm 1. The parameters λ_1, λ_2 are the two thresholds associated with the application of a pseudoexact algorithm in the main heuristic. Parameter C&P indicates the external pruning used in the pseudoexact algorithm when applied to a subproblem with $|S'|$ vertices. A denotes a choice of $|S'|/2 + 5$ for $|S'| > 10$, B denotes a choice of $|S'|$, C denotes a choice of $|S'|/2$ for $|S'| > 10$, D denotes a choice of $|S'|/3 + 2$ for $|S'| > 15$, E denotes a choice of $|S'|/2$ for $|S'| > 20$ and F denotes a choice of $|S'|/2 + 5$ for $|S'| > 20$. In all cases the choice is otherwise $|S'|$. $Adjchoices$ denotes the sequence of values of the variable

Table 2 Tuning parameters for instances that do require tabu search to find the best solution. Instances 1et.2048 and 1tc.2048 are split into subproblems as in the last paragraph of Section **??** and so are excluded from this table

Instance	θ_1	θ_2	θ_3	$\theta_{\rm ts}$	λ_1	λ_2	C&P	$adjchoices$	$tstenure$	$tstime$	$tssetmin$
(10,4) (Smith & Montemanni 2012)	182	205	210	205	105	45	A	S	6	720	195
(7,5) (Smith & Montemanni 2012)	66	80	80	66	125	85	A	S	6	400	52
C2000_9 (DIMACS 1993b)	70	74	76	70	105	45	E	S	12	5400	50
C4000_5 (DIMACS 1993b)	17	17	17	17	400	105	A	S	6	60	5
keller6 (DIMACS 1993b)	52	52	53	52	105	85	A	S	12	720	40
1dc.1024 (Sloane 2015)	78	78	78	78	105	75	A	S	20	360	65
1dc.2048 (Sloane 2015)	135	140	170	135	100	80	A	S	20	720	125
1et.1024 (Sloane 2015)	161	161	170	161	90	80	A	S	20	2880	156
1tc.1024 (Sloane 2015)	190	190	200	190	90	80	A	S	20	160	182
1zc.1024 (Sloane 2015)	107	110	111	107	95	85	C	S	12	60	12
1zc.2048 (Sloane 2015)	183	196	196	184	80	70	D	S	24	1440	177
2dc.2048 (Sloane 2015)	22	25	25	22	90	80	A	S	20	120	14
frb30-15-1 (BHOSLIB 2015)	24	29	29	24	85	25	F	S	12	20	16
frb30-15-2 (BHOSLIB 2015)	24	29	29	24	85	25	F	S	12	20	16
frb30-15-3 (BHOSLIB 2015)	24	29	29	24	85	25	F	S	12	20	16
frb30-15-4 (BHOSLIB 2015)	24	29	29	24	85	25	F	S	12	20	16
frb30-15-5 (BHOSLIB 2015)	24	29	29	24	85	25	F	S	12	20	16
frb35-17-1 (BHOSLIB 2015)	29	34	34	29	85	25	F	S	12	20	21
frb35-17-2 (BHOSLIB 2015)	29	34	34	29	85	25	F	S	12	20	21
frb35-17-3 (BHOSLIB 2015)	29	34	34	29	85	25	F	S	12	20	21
frb35-17-4 (BHOSLIB 2015)	29	34	34	29	85	25	F	S	12	20	21
frb35-17-5 (BHOSLIB 2015)	29	34	34	29	85	25	F	S	12	20	21
frb50-23-1 (BHOSLIB 2015)	44	49	49	44	85	25	F	S	12	20	36
frb50-23-2 (BHOSLIB 2015)	44	49	49	44	85	25	F	S	12	20	36
frb50-23-3 (BHOSLIB 2015)	44	49	49	44	85	25	F	S	12	20	36
frb50-23-4 (BHOSLIB 2015)	44	49	49	44	85	25	F	S	12	20	36
frb50-23-5 (BHOSLIB 2015)	44	49	49	44	85	25	F	S	12	20	36

$adjchoices$ cycled through by the main heuristic. The standard choice (denoted S in the table) is the sequence [1,50,80,100,120,160,200,300,600,800]. However, this is clearly not appropriate for the smaller Brock instances, and for these instances a fixed value of 1 for $adjchoices$ was found to be satisfactory (denoted ONE in the tables).

Although there appear to be many parameters to be set, there is a straightforward rationale for setting them based on some initial short runs. Initially, $\theta_{\rm ts}$ is set high to avoid tabu search and θ_1, θ_2, θ_3 are set low. These last three parameters are then raised until only good solutions are generated, but not so high that solutions are only very rarely pursued

further. For very large cliques 3_optimization may be very slow and θ_3 should be set very large in such cases. The parameters λ_1, λ_2 are set so that pseudoexact search does not slow the algorithm too much. Normally λ_1 is set somewhat larger that λ_2, as pseudoexact search with threshold λ_1 is applied much less frequently than pseudoexact search with threshold λ_2. Then $\theta_{\rm ts}$ can be set so that the most promising solutions are pursued by tabu search. There are three other parameters associated with tabu search. The parameter $tstenure$ is the length of the tabu list. The algorithm does not seem to be particularly sensitive to choices between 6 and 24, and several choices have been used. The parameter $tssetmin$ is associated with the size of the current clique before vertices are added. If this is too small the size of the subproblem to which pseudoexact search is applied is large and pseudoexact search is too slow. On the other hand, small values give a larger probability of finding new best solutions. Finally, $tstime$ describes the time for which tabu search is applied without improvement. Large values give longer run times but a larger probability of finding new best solutions, so the value chosen depends on the difficulty of the problem.

In order to demonstrate that some of the parameter tuning is not particularly critical, the 15 BHOSLIB instances were run with many of the parameters fixed. Thus λ_1, λ_2, C&P, $adjchoices$, $tstenure$ and $tstime$ were fixed. The other parameters bear a fixed relationship to the size of clique likely to be found, as indicated by a short run of the main heuristic.

Results for permutation code and hard DIMACS instances are given in Table **??**, results for Sloane code instances are given in Table **??** and results for BHOSLIB instances are given in Table **??**. Column 1 gives the instance, the source, the number of vertices and the number of edges. Column 2 gives the value of $adjchoices$ used for the first best solution, the method or methods used after the main heuristic for this solution, and the number of times a clique with the maximum number of vertices is found. Column 3 gives the number of vertices in the best clique found and the best known value previously. Optimal values (according to (Jin and Hao, 2015; Sloane, 2015)) are marked with *. Column 4 gives the time to find the first best clique and column 5 gives the total search time.

For the instance (7,5) the algorithm improves the best result obtained with existing algorithms, although it does not achieve the lower bound of 78 obtained in (Janiszczak et al., 2015) using a group theory construction. This instance appears to be a useful benchmark to add to those in the literature. The best known result is obtained in all other instances except C2000_9, where the best known lower bound is 80. This random graph instance is known to be particularly hard. For example, in (Jin and Hao, 2015) a clique of size 80 was only found twice in 100 runs of 10^8 iterations (see also (Grosso et al., 2008)). The frequency with which cliques of size 78 and 79 were found in testing suggests that cliques of size 80 might be reliably found only if run times were measured in weeks.

A comparison of solution quality and run times for 21 effective heuristics on the 9 hard DIMACS instances is given by Wu and Hao (2015) and hence is not repeated here. The solution quality of HTS presented here is better than all these heuristics except SBTS (Jin and Hao, 2015) (who find a clique of size 80 for C2000_9). Comparing run times, HTS is faster than SBTS on 4 DIMACS instances, but slower (sometimes much slower) on the others.

6 Conclusions

The main novel features of the algorithm HTS are (i) the use of a pseudoexact algorithm based on external pruning and enhanced pruning, (ii) the generation of starting solutions

Table 3 Results for permutation code and hard DIMACS instances

Instance	**Method**	**No. vertices in largest clique found**	**Time to find largest clique (s)**	**Time for complete search (s)**
(11,5) (Smith & Montemanni 2012)	$adjchoices$ = 120	26	3	20
$\|V\| = 3170$	- (10)			
$\|E\| = 3132680$	Best known	26	-	-
(12,4) (Smith & Montemanni 2012)	$adjchoices$ = 120	220	430	2566
$\|V\| = 4534$	- (10)			
$\|E\| = 9243807$	Best known	220	-	-
(12,5) (Smith & Montemanni 2012)	$adjchoices$ = 100	25	1	47
$\|V\| = 3049$	- (10)			
$\|E\| = 2784552$	Best known	25	-	-
(10,4) (Smith & Montemanni 2012)	$adjchoices$ = 80	209	1257	13561
$\|V\| = 4755$	one_ts (3)			
$\|E\| = 10660530$	Best known	209	-	-
(7,5) (Smith & Montemanni 2012)	$adjchoices$ = 300	72	5330	44605
$\|V\| = 4634$	ts (1)			
$\|E\| = 9855027$	Best known (theory)	78	-	-
	Algorithm	62		5700
brock400_2 (DIMACS 1993b)	$adjchoices$ = 1	29	589	3664
$\|V\| = 400$	- (4)			
$\|E\| = 59786$	Best known	29^*	-	-
brock400_4 (DIMACS 1993b)	$adjchoices$ = 1	33	32	319
$\|V\| = 400$	- (9)			
$\|E\| = 59765$	Best known	33^*	-	-
brock800_2 (DIMACS 1993b)	$adjchoices$ = 1	24	71	9864
$\|V\| = 800$	- (2)			
$\|E\| = 208166$	Best known	24^*	-	-
brock800_4 (DIMACS 1993b)	$adjchoices$ = 1	26	706	4377
$\|V\| = 800$	- (4)			
$\|E\| = 207643$	Best known	26^*	-	-
C2000_9 (DIMACS 1993b)	$adjchoices$ = 100	79	8920	45196
$\|V\| = 2000$	ts (1)			
$\|E\| = 1799532$	Best known	80	-	-
C4000_5 (DIMACS 1993b)	$adjchoices$ = 200	18	390	14271
$\|V\| = 4000$	ts (5)			
$\|E\| = 4000268$	Best known	18^*	-	-
keller6 (DIMACS 1993b)	$adjchoices$ = 600	59	197862	320777
$\|V\| = 3361$	ts (3)			
$\|E\| = 4000268$	Best known	59	-	-
MANN_a45 (DIMACS 1993b)	$adjchoices$ = 100	345	171	352
$\|V\| = 1035$	- (2)			
$\|E\| = 533115$	Best known	345^*	-	-
MANN_a81 (DIMACS 1993b)	$adjchoices$ = 100	1100	9	25
$\|V\| = 3321$	- (2)			
$\|E\| = 5506380$	Best known	1100^*	-	-

Table 4 Results for Sloane code instances with 1024 and 2048 vertices

Instance	Method	No. vertices in largest clique found	Time to find largest clique (s)	Time for complete search (s)
1dc.1024 (Sloane 2015)	$adjchoices$ = 800	94	11899	25099
$\|V\| = 1024$	ts (4)			
$\|E\| = 499713$	Best known	94^*	-	-
1dc.2048 (Sloane 2015)	$adjchoices$ = 600	172	20561	91425
$\|V\| = 2048$	ts (2)			
$\|E\| = 2037761$	Best known	172	-	-
1et.1024 (Sloane 2015)	$adjchoices$ = 160	171	25988	510191
$\|V\| = 1024$	ts (4)			
$\|E\| = 514176$	Best known	171^*	-	-
1et.2048 (Sloane 2015)	$adjchoices$ = 120	361	1366	1366
$\|V\| = 2048$	ts (1)			
$\|E\| = 2073600$	Best known	361^*	-	-
1tc.1024 (Sloane 2015)	$adjchoices$ = 600	196	1309	3501
$\|V\| = 1024$	ts (5)			
$\|E\| = 515840$	Best known	196^*	-	-
1tc.2048 (Sloane 2015)	$adjchoices$ = 100	352	797	797
$\|V\| = 2048$	ts (1)			
$\|E\| = 2077184$	Best known	352^*	-	-
1zc.1024 (Sloane 2015)	$adjchoices$ = 300	112	8880	16746
$\|V\| = 1024$	ts (1)			
$\|E\| = 507136$	Best known	112	-	-
1zc.2048 (Sloane 2015)	$adjchoices$ = 800	198	479971	500814
$\|V\| = 2048$	ts (1)			
$\|E\| = 2056704$	Best known	198	-	-
2dc.1024 (Sloane 2015)	$adjchoices$ = 800	16	51	241
$\|V\| = 1024$	- (10)			
$\|E\| = 354614$	Best known	16^*	-	-
2dc.2048 (Sloane 2015)	$adjchoices$ = 600	24	131	684
$\|V\| = 2048$	ts (3)			
$\|E\| = 1591677$	Best known	24^*	-	-

using the set S, (iii) a novel tabu search and (iv) preprocessing for some instances. In particular, the use of a pseudoexact algorithm is an attempt to undertake a more thorough local search of the solution space, at a cost of a much smaller number of iterations per second than some more conventional algorithms. The algorithm has been shown to be successful, although even with careful tuning HTS is significantly slower than many existing algorithms for some of the instances. In part this is due to the computers used and the relatively unsophisticated implementation. One of the main advantages of HTS may be the relative ease with which it can be implemented. It is worth noting that the best solution for

Table 5 Results for BHOSLIB instances with 450, 595 and 1150 vertices

Problem	**Method**	**No. vertices in largest clique found**	**Time to find largest clique (s)**	**Time for complete search (s)**
frb30-15-1 (BHOSLIB 2015)	adjchoices = 100	30	40	174
$\|V\| = 450$	ts (4)			
$\|E\| = 83198$	Best known	30^*	-	-
frb30-15-2 (BHOSLIB 2015)	adjchoices = 100	30	23	171
$\|V\| = 450$	one_ts (7)			
$\|E\| = 83151$	Best known	30^*	-	-
frb30-15-3 (BHOSLIB 2015)	adjchoices = 100	30	36	200
$\|V\| = 450$	ts (4)			
$\|E\| = 83216$	Best known	30^*	-	-
frb30-15-4 (BHOSLIB 2015)	adjchoices = 100	30	34	246
$\|V\| = 450$	ts (10)			
$\|E\| = 83194$	Best known	30^*	-	-
frb30-15-5 (BHOSLIB 2015)	adjchoices = 300	30	58	217
$\|V\| = 450$	ts (4)			
$\|E\| = 83231$	Best known	30^*	-	-
frb35-17-1 (BHOSLIB 2015)	adjchoices = 160	35	195	200
$\|V\| = 595$	ts (1)			
$\|E\| = 148859$	Best known	35^*	-	-
frb35-17-2 (BHOSLIB 2015)	adjchoices = 120	35	27	30
$\|V\| = 595$	ts (1)			
$\|E\| = 148868$	Best known	35^*	-	-
frb35-17-3 (BHOSLIB 2015)	adjchoices = 120	35	49	90
$\|V\| = 595$	ts (2)			
$\|E\| = 148784$	Best known	35^*	-	-
frb35-17-4 (BHOSLIB 2015)	adjchoices = 80	35	827	900
$\|V\| = 595$	ts (1)			
$\|E\| = 148873$	Best known	35^*	-	-
frb35-17-5 (BHOSLIB 2015)	adjchoices = 300	35	158	271
$\|V\| = 595$	ts (2)			
$\|E\| = 148572$	Best known	35^*	-	-
frb50-23-1 (BHOSLIB 2015)	adjchoices = 800	50	22903	72977
$\|V\| = 1150$	ts (1)			
$\|E\| = 580603$	Best known	50^*	-	-
frb50-23-2 (BHOSLIB 2015)	adjchoices = 800	50	90773	148223
$\|V\| = 1150$	ts (1)			
$\|E\| = 579824$	Best known	50^*	-	-
frb50-23-3 (BHOSLIB 2015)	adjchoices = 600	50	196409	417821
$\|V\| = 1150$	ts (1)			
$\|E\| = 579607$	Best known	50^*	-	-
frb50-23-4 (BHOSLIB 2015)	adjchoices = 600	50	139	26163
$\|V\| = 1150$	ts (6)			
$\|E\| = 580417$	Best known	50^*	-	-
frb50-23-5 (BHOSLIB 2015)	adjchoices = 200	50	583	74686
$\|V\| = 1150$	ts (2)			
$\|E\| = 580640$	Best known	50^*	-	-

10 of the 39 instances (including 4 hard DIMACS instances) can be found with the easily implemented main heuristic alone, without recourse to tabu search.

References

BHOSLIB (2015) Benchmarks with hidden optimum solutions for graph problems. http://iridia.ulb.ac.be/∼fmascia/maximum_clique/BHOSLIB-benchmark (Accessed 25 January 2017).

Carraghan, R. and Pardalos, P.M. (1990) 'An exact algorithm for the maximum clique problem', *Operations Research Letters*, Vol. 9 No. 6, pp.375–382

DIMACS (1993a) The second DIMACS implementation challenge: 1992-1993. http://dimacs.rutgers.edu/Challenges/ (Accessed 25 January 2017).

DIMACS (1993b) Clique benchmark instances. https://turing.cs.hbg.psu.edu/txn131/clique.html (Accessed 25 January 2017).

Gurobi (2017) Gurobi optimizer. http://www.gurobi.com (Accessed 15 November 2017).

Gendreau, M., Soriano, P. and Salvail, L. (1993) 'Solving the maximum clique algorithm using a tabu search approach', *Annals of Operations Research*, Vol. 41 No. 4, pp.385–403

Grosso, A., Locatelli, M. and Pullan, W. (2008) 'Simple ingredients leading to very efficient heuristics for the maximum clique problem', *Journal of Heuristics*, Vol. 14 No.6, pp.587–612

Janiszczak, I., Lempken, W., Östergård, P.R.J. and Staszewski R. (2015) 'Permutation codes invariant under isometries', *Designs, Codes and Cryptography*, Vol. 75 No. 3, pp.497—507

Jin, Y., Hao, J-K. (2015) 'General swap-based neighborhood tabu search for the maximum independent set problem', *Engineering Applications of Artificial Intelligence*. Vol. 37, pp.20–33

Leese, R., Hurley, S. (Eds.) (2002) *Methods and Algorithms for Radio Channel Assignment*, Oxford University Press, Oxford.

Magma Handbook (2017) Graphs. https://magma.maths.usyd.edu.au/magma/handbook/graphs (Accessed 17 October 2017).

McCreesh, C. and Prosser, P. (2013) 'Multi-threading a state-of-the-art maximum clique algorithm', *Algorithms*, Vol. 6 No. 4, pp.618–635

Sloane, N.J.A. (2015) Challenge problems: independent sets in graphs. https://oeis.org/a265032.html (Accessed 25 January 2017).

Smith, D.H., Montemanni, R. (2012) 'A new table of permutation codes', *Designs, Codes and Cryptography*, Vol. 63 No. 2, pp.241–253

Wu, Q. and Hao, J-K. (2015) 'A review on algorithms for maximum clique problems', *European Journal of Operational Research*, Vol. 242, No. 3, pp.693–709

Wu, Q., Hao, J-K. and Glover, F. (2012) 'Multi-neighborhood tabu search for the maximum weight clique problem', *Annals of Operations Research*, Vol. 196 No. 1, pp.611–634